UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/675,745	09/30/2003	Manu Gulati	BP3252	8011

51472 7590 06/11/2009

GARLICK HARRISON & MARKISON
P.O. BOX 160727
AUSTIN, TX 78716-0727

EXAMINER	
AHMED, SALMAN	
ART UNIT	PAPER NUMBER
2419	
MAIL DATE	DELIVERY MODE
06/11/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/675,745	GULATI ET AL.	
	Examiner	Art Unit	
	SALMAN AHMED	2419	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *07 April 2009*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-29* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-29* is/are rejected.

7)☒ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *30 August 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Claims 1-29 are pending.

Claims 1-29 are rejected.

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

3. Claims 1-29 are rejected under 35 U.S.C. 103(a) as being unpatentable over Oberman et al., hereinafter Oberman, (US7042891) in view of Traw et al. (US PAT 5274768, hereinafter Traw).

Regarding claim 1, Oberman discloses dynmic selection of lowest latency path in a network switch (see Oberman col. 2 lines 29 - 58) comprising: • receiving a data block

at a receiver of the host device (see Oberman col. 2 lines 59-62); • storing the data block in a receiver buffer (see Oberman col. 7 lines 32-35); • determining an input virtual channel corresponding to the data block (see Oberman col. 8 lines 11-15); • updating an input virtual channel linked list corresponding to the input virtual channel to include the data block (see Oberman col. 7 lines 29-57) • determining an output virtual channel for the data block (see Oberman col. 8 lines 11-15); • transferring the data block from the input virtual channel linked list of the receiver buffer to a destination within the host device via the output virtual channel (see Oberman col. 8 lines 20-33); • updating the input virtual channel linked list to remove the data block (see Oberman col. 9 line 65).

Oberman implicitly teaches updating an input virtual channel linked list corresponding to the input virtual channel to include the data block but does not explicitly teach updating an input virtual channel linked list corresponding to the input virtual channel to include the data block; storing the data block in the receiver buffer includes storing the data block in the receiver buffer at an old free linked list head address ; the host device via the output virtual channel, wherein transferring the data block from the input virtual channel linked list of the receiver buffer to a destination within the host device via the output virtual channel includes reading the data block from the receiver buffer at an old input virtual channel linked list head address.

Traw in the same or similar field of endeavor teaches updating an input virtual channel linked list corresponding to the input virtual channel to include the data block; storing the data block in the receiver buffer includes storing the data block in the receiver buffer at an old free linked list head address (columns 5-6, lines 65-11, the use

of linked list data structures allows memory to be dynamically assigned to incoming ATM cells and provides a first in-first out (FIFO) queue for the cells of a particular VCI. The memory allocation is dynamic because incoming cells are placed into nodes that are removed from a free list (discussed below) and added to the linked list for the particular VCI. In this way, VCIs that are more active are allotted more memory than less active VCIs. This is accomplished by transferring the node back to the free list. The FIFO queuing characteristic of the linked list structure is important because the order of cells of a particular VCI must be maintained); the host device via the output virtual channel, wherein transferring the data block from the input virtual channel linked list of the receiver buffer to a destination within the host device via the output virtual channel includes reading the data block from the receiver buffer at an old input virtual channel linked list head address (column 7 lines 34-58, the Linked List Controller 18 moves to the initiate command state (S33) when commands requesting operation on the linked list of a particular VCI are passed from the VCI Lookup Controller 52 (FIG. 5). Those commands include commands for removing a cell, writing (i.e., adding) a cell and reading a cell from a linked list for a particular VCI. The Linked List Controller 18 moves to state S34 when a remove command is issued In state S34 the linked list associated with the VCI to be removed is appended to the free list. The Linked List Controller 18 moves to state S35 when a read command is issued. In state S35 the Linked List Controller 18 locates the first cell reference structure (oldest cell) in the linked list associated with the VCI and determines the location of the cell body in the Reassembly Memory 44. The location of the cell body is passed to the Dual Port Controller 46 (FIG.

4) in state S39 and the count (number of cells in the VCI's linked list) is passed to the host 4 in state S40).

It would have been obvious to one having ordinary skill in the art at the time the invention was made to incorporate in Oberman's system/method the steps of updating an input virtual channel linked list corresponding to the input virtual channel to include the data block; storing the data block in the receiver buffer includes storing the data block in the receiver buffer at an old free linked list head address ; the host device via the output virtual channel, wherein transferring the data block from the input virtual channel linked list of the receiver buffer to a destination within the host device via the output virtual channel includes reading the data block from the receiver buffer at an old input virtual channel linked list head address as suggested by Traw. The motivation is that (as suggested by Traw, column 6 lines 12-24) the linked list data consists of pointers and cell counts; the cell bodies are stored separately in the Dual Port Reassembly Buffer; the cell body data is separated from the linked list data to minimize the amount of data movement; this separation is particularly important because it allows linked list management and data movement operations to be carried out in parallel; the separation of the linked list and cell body data will become even more important as the network speed is increased because it limits the degree to which memory bandwidth limitations can throttle the interface. Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces/market place incentives if the variations are predictable to one of ordinary skill in the art.

Regarding claim 2, Oberman teaches determining an output virtual channel for the data block includes processing one or more of the input virtual channel, a header corresponding to the data block (see Oberman col. 9 lines 25-67), a protocol corresponding to the data block (see Oberman col. 22 lines 23), source identifier/address corresponding to the data block (see Oberman col. 10 lines 43), and a destination identifier/address corresponding to the data block (see Oberman col. 8 lines 2).

Regarding claim 3, Oberman does not explicitly teach updating an input virtual channel linked list corresponding to the input virtual channel to include the data block includes: reading a new free linked list head address from the receiver buffer at an old free linked list head address; writing the new free linked list head address to a free linked list head register; writing the old free linked list head address to the receiver buffer at the old input virtual channel linked list tail address; and writing the old free linked list head address to an input virtual channel linked list tail register.

Traw in the same or similar field of endeavor teaches (columns 5-7, lines 62-29, FIG. 3 is a block diagram of the Linked List Manager 10. The Linked List Manager 10 is capable of performing a number of operations on the linked list data (depicted in FIG. 3B) to effect reassembly. The use of linked list data structures allows memory to be dynamically assigned to incoming ATM cells and provides a first in-first out (FIFO) queue for the cells of a particular VCI. The memory allocation is dynamic because incoming cells are placed into nodes that are removed from a free list (discussed below) and added to the linked list for the particular VCI. In this way, VCIs that are more active

are allotted more memory than less active VCIs. Moreover, memory may be deallocated as soon as a cell body has been read. This is accomplished by transferring the node back to the free list. The FIFO queuing characteristic of the linked list structure is important because the order of cells of a particular VCI must be maintained. The linked list data consists of pointers and cell counts. The cell bodies are stored separately in the Dual Port Reassembly Buffer 12, which is described below. The cell body data is separated from the linked list data to minimize the amount of data movement. This separation is particularly important because it allows linked list management and data movement operations to be carried out in parallel. The separation of the linked list and cell body data will become even more important as the network speed is increased because it limits the degree to which memory bandwidth limitations can throttle the interface. The Linked List Manager 10 receives references to and commands for the manipulation of a particular VCI from the VCI Lookup Controller 52 (which is part of block 14, FIG. 5). It also provides status information for the host 4 (FIG. 1) and addresses in the Reassembly Buffer 12 to which data is to be moved. The Linked List Manager 10 is composed of a Linked List Controller 18, buffers 20, 22 (used for configuration), counter 24, and registers 26, 28, 32, 34, 36, which are used in managing the linked lists, and a Pointer Table memory 30 of 8192 (8K) words, which is used to store the linked list data necessary for reassembly. The Linked List Controller 18, buffers 20, 22, counter 24, and registers 26, 28, 32, 34, 36 are provided by an EPM5128 programmable logic device in the preferred embodiment; the Pointer Table memory 30 is composed of two 8K by 48 MCM6164-45C static random access

memories (RAMs). All of the nodes are assembled into a linked list, known as the free list, during initialization. Operation of the host interface 1 is suspended during initialization, during which all pointers and counters used to keep track of the linked lists are zeroed. Initialization is performed by the host 4 by requesting a setup state and writing the appropriate configuration data to buffers 20, 22. The free list is a linked list of nodes that are not currently assigned to a VCI. The free list is defined by the values stored in locations 2048 and 2049 of the Pointer Table memory 30 (FIG. 3B), which represent the addresses of the first and last free nodes in the free node linked list. Four operations may be performed by the Linked List Manager 10 after initialization:

1. Cells may be added to the linked list for a particular VCI.
2. Cells may be removed from a VCI's linked list.
3. A VCI may be cleared, returning all of its nodes to the free list.
4. A cell count for a VCI may be obtained and returned to the host.

The first two operations pass the Reassembly Memory 44 address of the cell body that is to be affected to the Dual Port Controller 46 (44 and 46 part of the Dual Port Reassembly Buffer 12 and are shown in FIG. 4). The count and linked list associated with that VCI are also updated. A first portion (the first 1K (1024) words) of the Pointer Table memory 30 comprises a series of storage locations for each of the 256 possible VCIs. This structure is depicted in FIG. 3B. For each VCI, the pointer to the newest cell received (the last cell) is stored in a first location, the pointer to the oldest cell (first cell) is stored in a second location, and the current count of cells for the particular VCI is stored in a third location. The respective pointers point to locations in the linked lists,

which are located in the last 4K of the Pointer Table memory 30. It is useful to have access to a count of the number of cells available for a particular VCI in order to gauge the volume of traffic on that VCI. A second portion (the last 4K words, locations 4096-8192) of the Pointer Table memory 30 comprises sets of locations for storing pointers used for the nodes of the linked lists. Each node physically comprises two consecutive memory locations beginning on every even address in the 4K to 8K segment of the Pointer Table memory 30. The first location of each node stores the address in (or pointer to) the Reassembly Memory 44 (FIG. 4) where the cell body associated with that node is stored. The value of the first pointer is undefined if the node is in the empty list (i.e., if the node has not been allocated to a VCI). The second location of each node stores a pointer to the next node in the linked list. The value is zero if the node is the last node in the linked list.

It would have been obvious to one having ordinary skill in the art at the time the invention was made to incorporate in Oberman's system/method the steps of updating an input virtual channel linked list corresponding to the input virtual channel to include the data block includes: reading a new free linked list head address from the receiver buffer at an old free linked list head address; writing the new free linked list head address to a free linked list head register; writing the old free linked list head address to the receiver buffer at the old input virtual channel linked list tail address; and writing the old free linked list head address to an input virtual channel linked list tail register as suggested by Traw. The motivation is that (as suggested by Traw, column 6 lines 12-24) the linked list data consists of pointers and cell counts; the cell bodies are stored

separately in the Dual Port Reassembly Buffer; the cell body data is separated from the linked list data to minimize the amount of data movement; this separation is particularly important because it allows linked list management and data movement operations to be carried out in parallel; the separation of the linked list and cell body data will become even more important as the network speed is increased because it limits the degree to which memory bandwidth limitations can throttle the interface. Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces/market place incentives if the variations are predictable to one of ordinary skill in the art.

Regarding claim 4 Oberman does not explicitly teach updating the input virtual channel linked list to remove the data block includes: reading a new input virtual channel linked list head address from the receiver buffer at the old input virtual channel linked list head address; writing the new input virtual channel linked list head address to an input virtual channel linked list head register; writing the old input virtual channel linked list head address to the receiver buffer at an old free linked list tail address; and writing the old input virtual channel linked list head address to a free linked list tail register.

Traw in the same or similar field of endeavor teaches (columns 5-7, lines 62-29, FIG. 3 is a block diagram of the Linked List Manager 10. The Linked List Manager 10 is capable of performing a number of operations on the linked list data (depicted in FIG. 3B) to effect reassembly. The use of linked list data structures allows memory to be dynamically assigned to incoming ATM cells and provides a first in-first out (FIFO)

queue for the cells of a particular VCI. The memory allocation is dynamic because incoming cells are placed into nodes that are removed from a free list (discussed below) and added to the linked list for the particular VCI. In this way, VCIs that are more active are allotted more memory than less active VCIs. Moreover, memory may be deallocated as soon as a cell body has been read. This is accomplished by transferring the node back to the free list. The FIFO queuing characteristic of the linked list structure is important because the order of cells of a particular VCI must be maintained. The linked list data consists of pointers and cell counts. The cell bodies are stored separately in the Dual Port Reassembly Buffer 12, which is described below. The cell body data is separated from the linked list data to minimize the amount of data movement. This separation is particularly important because it allows linked list management and data movement operations to be carried out in parallel. The separation of the linked list and cell body data will become even more important as the network speed is increased because it limits the degree to which memory bandwidth limitations can throttle the interface. The Linked List Manager 10 receives references to and commands for the manipulation of a particular VCI from the VCI Lookup Controller 52 (which is part of block 14, FIG. 5). It also provides status information for the host 4 (FIG. 1) and addresses in the Reassembly Buffer 12 to which data is to be moved. The Linked List Manager 10 is composed of a Linked List Controller 18, buffers 20, 22 (used for configuration), counter 24, and registers 26, 28, 32, 34, 36, which are used in managing the linked lists, and a Pointer Table memory 30 of 8192 (8K) words, which is used to store the linked list data necessary for reassembly. The Linked List Controller

18, buffers 20, 22, counter 24, and registers 26, 28, 32, 34, 36 are provided by an EPM5128 programmable logic device in the preferred embodiment; the Pointer Table memory 30 is composed of two 8K by 48 MCM6164-45C static random access memories (RAMs). All of the nodes are assembled into a linked list, known as the free list, during initialization. Operation of the host interface 1 is suspended during initialization, during which all pointers and counters used to keep track of the linked lists are zeroed. Initialization is performed by the host 4 by requesting a setup state and writing the appropriate configuration data to buffers 20, 22. The free list is a linked list of nodes that are not currently assigned to a VCI. The free list is defined by the values stored in locations 2048 and 2049 of the Pointer Table memory 30 (FIG. 3B), which represent the addresses of the first and last free nodes in the free node linked list. Four operations may be performed by the Linked List Manager 10 after initialization:

1. Cells may be added to the linked list for a particular VCI.
2. Cells may be removed from a VCI's linked list.
3. A VCI may be cleared, returning all of its nodes to the free list.
4. A cell count for a VCI may be obtained and returned to the host.

The first two operations pass the Reassembly Memory 44 address of the cell body that is to be affected to the Dual Port Controller 46 (44 and 46 part of the Dual Port Reassembly Buffer 12 and are shown in FIG. 4). The count and linked list associated with that VCI are also updated. A first portion (the first 1K (1024) words) of the Pointer Table memory 30 comprises a series of storage locations for each of the 256 possible VCIs. This structure is depicted in FIG. 3B. For each VCI, the pointer to the newest cell

received (the last cell) is stored in a first location, the pointer to the oldest cell (first cell) is stored in a second location, and the current count of cells for the particular VCI is stored in a third location. The respective pointers point to locations in the linked lists, which are located in the last 4K of the Pointer Table memory 30. It is useful to have access to a count of the number of cells available for a particular VCI in order to gauge the volume of traffic on that VCI. A second portion (the last 4K words, locations 4096-8192) of the Pointer Table memory 30 comprises sets of locations for storing pointers used for the nodes of the linked lists. Each node physically comprises two consecutive memory locations beginning on every even address in the 4K to 8K segment of the Pointer Table memory 30. The first location of each node stores the address in (or pointer to) the Reassembly Memory 44 (FIG. 4) where the cell body associated with that node is stored. The value of the first pointer is undefined if the node is in the empty list (i.e., if the node has not been allocated to a VCI). The second location of each node stores a pointer to the next node in the linked list. The value is zero if the node is the last node in the linked list.

It would have been obvious to one having ordinary skill in the art at the time the invention was made to incorporate in Oberman's system/method the steps of updating the input virtual channel linked list to remove the data block includes: reading a new input virtual channel linked list head address from the receiver buffer at the old input virtual channel linked list head address; writing the new input virtual channel linked list head address to an input virtual channel linked list head register; writing the old input virtual channel linked list head address to the receiver buffer at an old free linked list tail

address; and writing the old input virtual channel linked list head address to a free linked list tail register as suggested by Traw. The motivation is that (as suggested by Traw, column 6 lines 12-24) the linked list data consists of pointers and cell counts; the cell bodies are stored separately in the Dual Port Reassembly Buffer; the cell body data is separated from the linked list data to minimize the amount of data movement; this separation is particularly important because it allows linked list management and data movement operations to be carried out in parallel; the separation of the linked list and cell body data will become even more important as the network speed is increased because it limits the degree to which memory bandwidth limitations can throttle the interface. Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces/market place incentives if the variations are predictable to one of ordinary skill in the art.

Regarding claim 7 Oberman does not explicitly teach in a common read/write cycle in which a first data block is read from the receiver buffer and a second data block is written to the receiver buffer: reading the first data block and a new input virtual channel head address from the receiver buffer at an old input virtual channel head address; writing the new input virtual channel head address to the input virtual channel head register; writing the second data block to the receiver buffer at the old input virtual channel head address; writing the old input virtual channel head address to an input virtual channel tail register; and writing the old input virtual channel head address to the receiver buffer at an old input virtual channel tail address.

Traw in the same or similar field of endeavor teaches (columns 5-7, lines 62-29, FIG. 3 is a block diagram of the Linked List Manager 10. The Linked List Manager 10 is capable of performing a number of operations on the linked list data (depicted in FIG. 3B) to effect reassembly. The use of linked list data structures allows memory to be dynamically assigned to incoming ATM cells and provides a first in-first out (FIFO) queue for the cells of a particular VCI. The memory allocation is dynamic because incoming cells are placed into nodes that are removed from a free list (discussed below) and added to the linked list for the particular VCI. In this way, VCIs that are more active are allotted more memory than less active VCIs. Moreover, memory may be deallocated as soon as a cell body has been read. This is accomplished by transferring the node back to the free list. The FIFO queuing characteristic of the linked list structure is important because the order of cells of a particular VCI must be maintained. The linked list data consists of pointers and cell counts. The cell bodies are stored separately in the Dual Port Reassembly Buffer 12, which is described below. The cell body data is separated from the linked list data to minimize the amount of data movement. This separation is particularly important because it allows linked list management and data movement operations to be carried out in parallel. The separation of the linked list and cell body data will become even more important as the network speed is increased because it limits the degree to which memory bandwidth limitations can throttle the interface. The Linked List Manager 10 receives references to and commands for the manipulation of a particular VCI from the VCI Lookup Controller 52 (which is part of block 14, FIG. 5). It also provides status information for the host 4

(FIG. 1) and addresses in the Reassembly Buffer 12 to which data is to be moved. The Linked List Manager 10 is composed of a Linked List Controller 18, buffers 20, 22 (used for configuration), counter 24, and registers 26, 28, 32, 34, 36, which are used in managing the linked lists, and a Pointer Table memory 30 of 8192 (8K) words, which is used to store the linked list data necessary for reassembly. The Linked List Controller 18, buffers 20, 22, counter 24, and registers 26, 28, 32, 34, 36 are provided by an EPM5128 programmable logic device in the preferred embodiment; the Pointer Table memory 30 is composed of two 8K by 48 MCM6164-45C static random access memories (RAMs). All of the nodes are assembled into a linked list, known as the free list, during initialization. Operation of the host interface 1 is suspended during initialization, during which all pointers and counters used to keep track of the linked lists are zeroed. Initialization is performed by the host 4 by requesting a setup state and writing the appropriate configuration data to buffers 20, 22. The free list is a linked list of nodes that are not currently assigned to a VCI. The free list is defined by the values stored in locations 2048 and 2049 of the Pointer Table memory 30 (FIG. 3B), which represent the addresses of the first and last free nodes in the free node linked list. Four operations may be performed by the Linked List Manager 10 after initialization:

1. Cells may be added to the linked list for a particular VCI.
2. Cells may be removed from a VCI's linked list.
3. A VCI may be cleared, returning all of its nodes to the free list.
4. A cell count for a VCI may be obtained and returned to the host.

The first two operations pass the Reassembly Memory 44 address of the cell body that is to be affected to the Dual Port Controller 46 (44 and 46 part of the Dual Port Reassembly Buffer 12 and are shown in FIG. 4). The count and linked list associated with that VCI are also updated. A first portion (the first 1K (1024) words) of the Pointer Table memory 30 comprises a series of storage locations for each of the 256 possible VCIs. This structure is depicted in FIG. 3B. For each VCI, the pointer to the newest cell received (the last cell) is stored in a first location, the pointer to the oldest cell (first cell) is stored in a second location, and the current count of cells for the particular VCI is stored in a third location. The respective pointers point to locations in the linked lists, which are located in the last 4K of the Pointer Table memory 30. It is useful to have access to a count of the number of cells available for a particular VCI in order to gauge the volume of traffic on that VCI. A second portion (the last 4K words, locations 4096-8192) of the Pointer Table memory 30 comprises sets of locations for storing pointers used for the nodes of the linked lists. Each node physically comprises two consecutive memory locations beginning on every even address in the 4K to 8K segment of the Pointer Table memory 30. The first location of each node stores the address in (or pointer to) the Reassembly Memory 44 (FIG. 4) where the cell body associated with that node is stored. The value of the first pointer is undefined if the node is in the empty list (i.e., if the node has not been allocated to a VCI). The second location of each node stores a pointer to the next node in the linked list. The value is zero if the node is the last node in the linked list.

It would have been obvious to one having ordinary skill in the art at the time the invention was made to incorporate in Oberman's system/method the steps of in a common read/write cycle in which a first data block is read from the receiver buffer and a second data block is written to the receiver buffer: reading the first data block and a new input virtual channel head address from the receiver buffer at an old input virtual channel head address; writing the new input virtual channel head address to the input virtual channel head register; writing the second data block to the receiver buffer at the old input virtual channel head address; writing the old input virtual channel head address to an input virtual channel tail register; and writing the old input virtual channel head address to the receiver buffer at an old input virtual channel tail address as suggested by Traw. The motivation is that (as suggested by Traw, column 6 lines 12-24) the linked list data consists of pointers and cell counts; the cell bodies are stored separately in the Dual Port Reassembly Buffer; the cell body data is separated from the linked list data to minimize the amount of data movement; this separation is particularly important because it allows linked list management and data movement operations to be carried out in parallel; the separation of the linked list and cell body data will become even more important as the network speed is increased because it limits the degree to which memory bandwidth limitations can throttle the interface. Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces/market place incentives if the variations are predictable to one of ordinary skill in the art.

Regarding claim 13 Oberman does not explicitly teach updating an input virtual channel linked list corresponding to the input virtual channel to include the data block comprises: reading a new free linked list head address from the receiver buffer at an old free linked list head address; writing the new free linked list head address to a free linked list head register; writing the old free linked list head address to the receiver buffer at the old input virtual channel linked list tail address; and writing the old free linked list head address to an input virtual channel linked list tail register.

Traw in the same or similar field of endeavor teaches (columns 5-7, lines 62-29, FIG. 3 is a block diagram of the Linked List Manager 10. The Linked List Manager 10 is capable of performing a number of operations on the linked list data (depicted in FIG. 3B) to effect reassembly. The use of linked list data structures allows memory to be dynamically assigned to incoming ATM cells and provides a first in-first out (FIFO) queue for the cells of a particular VCI. The memory allocation is dynamic because incoming cells are placed into nodes that are removed from a free list (discussed below) and added to the linked list for the particular VCI. In this way, VCIs that are more active are allotted more memory than less active VCIs. Moreover, memory may be deallocated as soon as a cell body has been read. This is accomplished by transferring the node back to the free list. The FIFO queuing characteristic of the linked list structure is important because the order of cells of a particular VCI must be maintained. The linked list data consists of pointers and cell counts. The cell bodies are stored separately in the Dual Port Reassembly Buffer 12, which is described below. The cell body data is separated from the linked list data to minimize the amount of data

movement. This separation is particularly important because it allows linked list management and data movement operations to be carried out in parallel. The separation of the linked list and cell body data will become even more important as the network speed is increased because it limits the degree to which memory bandwidth limitations can throttle the interface. The Linked List Manager 10 receives references to and commands for the manipulation of a particular VCI from the VCI Lookup Controller 52 (which is part of block 14, FIG. 5). It also provides status information for the host 4 (FIG. 1) and addresses in the Reassembly Buffer 12 to which data is to be moved. The Linked List Manager 10 is composed of a Linked List Controller 18, buffers 20, 22 (used for configuration), counter 24, and registers 26, 28, 32, 34, 36, which are used in managing the linked lists, and a Pointer Table memory 30 of 8192 (8K) words, which is used to store the linked list data necessary for reassembly. The Linked List Controller 18, buffers 20, 22, counter 24, and registers 26, 28, 32, 34, 36 are provided by an EPM5128 programmable logic device in the preferred embodiment; the Pointer Table memory 30 is composed of two 8K by 48 MCM6164-45C static random access memories (RAMs). All of the nodes are assembled into a linked list, known as the free list, during initialization. Operation of the host interface 1 is suspended during initialization, during which all pointers and counters used to keep track of the linked lists are zeroed. Initialization is performed by the host 4 by requesting a setup state and writing the appropriate configuration data to buffers 20, 22. The free list is a linked list of nodes that are not currently assigned to a VCI. The free list is defined by the values stored in locations 2048 and 2049 of the Pointer Table memory 30 (FIG. 3B), which

represent the addresses of the first and last free nodes in the free node linked list. Four operations may be performed by the Linked List Manager 10 after initialization:

1. Cells may be added to the linked list for a particular VCI.
2. Cells may be removed from a VCI's linked list.
3. A VCI may be cleared, returning all of its nodes to the free list.
4. A cell count for a VCI may be obtained and returned to the host.

The first two operations pass the Reassembly Memory 44 address of the cell body that is to be affected to the Dual Port Controller 46 (44 and 46 part of the Dual Port Reassembly Buffer 12 and are shown in FIG. 4). The count and linked list associated with that VCI are also updated. A first portion (the first 1K (1024) words) of the Pointer Table memory 30 comprises a series of storage locations for each of the 256 possible VCIs. This structure is depicted in FIG. 3B. For each VCI, the pointer to the newest cell received (the last cell) is stored in a first location, the pointer to the oldest cell (first cell) is stored in a second location, and the current count of cells for the particular VCI is stored in a third location. The respective pointers point to locations in the linked lists, which are located in the last 4K of the Pointer Table memory 30. It is useful to have access to a count of the number of cells available for a particular VCI in order to gauge the volume of traffic on that VCI. A second portion (the last 4K words, locations 4096-8192) of the Pointer Table memory 30 comprises sets of locations for storing pointers used for the nodes of the linked lists. Each node physically comprises two consecutive memory locations beginning on every even address in the 4K to 8K segment of the Pointer Table memory 30. The first location of each node stores the address in (or

pointer to) the Reassembly Memory 44 (FIG. 4) where the cell body associated with that node is stored. The value of the first pointer is undefined if the node is in the empty list (i.e., if the node has not been allocated to a VCI). The second location of each node stores a pointer to the next node in the linked list. The value is zero if the node is the last node in the linked list.

It would have been obvious to one having ordinary skill in the art at the time the invention was made to incorporate in Oberman's system/method the steps of updating an input virtual channel linked list corresponding to the input virtual channel to include the data block comprises: reading a new free linked list head address from the receiver buffer at an old free linked list head address; writing the new free linked list head address to a free linked list head register; writing the old free linked list head address to the receiver buffer at the old input virtual channel linked list tail address; and writing the old free linked list head address to an input virtual channel linked list tail register as suggested by Traw. The motivation is that (as suggested by Traw, column 6 lines 12-24) the linked list data consists of pointers and cell counts; the cell bodies are stored separately in the Dual Port Reassembly Buffer; the cell body data is separated from the linked list data to minimize the amount of data movement; this separation is particularly important because it allows linked list management and data movement operations to be carried out in parallel; the separation of the linked list and cell body data will become even more important as the network speed is increased because it limits the degree to which memory bandwidth limitations can throttle the interface. Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one

based on design incentives or other market forces/market place incentives if the variations are predictable to one of ordinary skill in the art.

Regarding claim 16 Oberman does not explicitly teach in a common read/write cycle in which a first data block is read from the receiver buffer and a second data block is written to the receiver buffer: reading the first data block and a new output virtual channel head address from the receiver buffer at the old output virtual channel head address; writing the new output virtual channel head address to the output virtual channel head register; writing the second data block to the receiver buffer at the old output virtual channel head address; writing the old output virtual channel head address to an output virtual channel tail register; and writing the old output virtual channel head address to the receiver buffer at the old output virtual channel head address.

Traw in the same or similar field of endeavor teaches (columns 5-7, lines 62-29, FIG. 3 is a block diagram of the Linked List Manager 10. The Linked List Manager 10 is capable of performing a number of operations on the linked list data (depicted in FIG. 3B) to effect reassembly. The use of linked list data structures allows memory to be dynamically assigned to incoming ATM cells and provides a first in-first out (FIFO) queue for the cells of a particular VCI. The memory allocation is dynamic because incoming cells are placed into nodes that are removed from a free list (discussed below) and added to the linked list for the particular VCI. In this way, VCIs that are more active are allotted more memory than less active VCIs. Moreover, memory may be deallocated as soon as a cell body has been read. This is accomplished by transferring the node back to the free list. The FIFO queuing characteristic of the linked list

structure is important because the order of cells of a particular VCI must be maintained. The linked list data consists of pointers and cell counts. The cell bodies are stored separately in the Dual Port Reassembly Buffer 12, which is described below. The cell body data is separated from the linked list data to minimize the amount of data movement. This separation is particularly important because it allows linked list management and data movement operations to be carried out in parallel. The separation of the linked list and cell body data will become even more important as the network speed is increased because it limits the degree to which memory bandwidth limitations can throttle the interface. The Linked List Manager 10 receives references to and commands for the manipulation of a particular VCI from the VCI Lookup Controller 52 (which is part of block 14, FIG. 5). It also provides status information for the host 4 (FIG. 1) and addresses in the Reassembly Buffer 12 to which data is to be moved. The Linked List Manager 10 is composed of a Linked List Controller 18, buffers 20, 22 (used for configuration), counter 24, and registers 26, 28, 32, 34, 36, which are used in managing the linked lists, and a Pointer Table memory 30 of 8192 (8K) words, which is used to store the linked list data necessary for reassembly. The Linked List Controller 18, buffers 20, 22, counter 24, and registers 26, 28, 32, 34, 36 are provided by an EPM5128 programmable logic device in the preferred embodiment; the Pointer Table memory 30 is composed of two 8K by 48 MCM6164-45C static random access memories (RAMs). All of the nodes are assembled into a linked list, known as the free list, during initialization. Operation of the host interface 1 is suspended during initialization, during which all pointers and counters used to keep track of the linked lists

are zeroed. Initialization is performed by the host 4 by requesting a setup state and writing the appropriate configuration data to buffers 20, 22. The free list is a linked list of nodes that are not currently assigned to a VCI. The free list is defined by the values stored in locations 2048 and 2049 of the Pointer Table memory 30 (FIG. 3B), which represent the addresses of the first and last free nodes in the free node linked list. Four operations may be performed by the Linked List Manager 10 after initialization:

1. Cells may be added to the linked list for a particular VCI.
2. Cells may be removed from a VCI's linked list.
3. A VCI may be cleared, returning all of its nodes to the free list.
4. A cell count for a VCI may be obtained and returned to the host.

The first two operations pass the Reassembly Memory 44 address of the cell body that is to be affected to the Dual Port Controller 46 (44 and 46 part of the Dual Port Reassembly Buffer 12 and are shown in FIG. 4). The count and linked list associated with that VCI are also updated. A first portion (the first 1K (1024) words) of the Pointer Table memory 30 comprises a series of storage locations for each of the 256 possible VCIs. This structure is depicted in FIG. 3B. For each VCI, the pointer to the newest cell received (the last cell) is stored in a first location, the pointer to the oldest cell (first cell) is stored in a second location, and the current count of cells for the particular VCI is stored in a third location. The respective pointers point to locations in the linked lists, which are located in the last 4K of the Pointer Table memory 30. It is useful to have access to a count of the number of cells available for a particular VCI in order to gauge the volume of traffic on that VCI. A second portion (the last 4K words, locations 4096-

8192) of the Pointer Table memory 30 comprises sets of locations for storing pointers used for the nodes of the linked lists. Each node physically comprises two consecutive memory locations beginning on every even address in the 4K to 8K segment of the Pointer Table memory 30. The first location of each node stores the address in (or pointer to) the Reassembly Memory 44 (FIG. 4) where the cell body associated with that node is stored. The value of the first pointer is undefined if the node is in the empty list (i.e., if the node has not been allocated to a VCI). The second location of each node stores a pointer to the next node in the linked list. The value is zero if the node is the last node in the linked list.

It would have been obvious to one having ordinary skill in the art at the time the invention was made to incorporate in Oberman's system/method the steps of in a common read/write cycle in which a first data block is read from the receiver buffer and a second data block is written to the receiver buffer: reading the first data block and a new output virtual channel head address from the receiver buffer at the old output virtual channel head address; writing the new output virtual channel head address to the output virtual channel head register; writing the second data block to the receiver buffer at the old output virtual channel head address; writing the old output virtual channel head address to an output virtual channel tail register; and writing the old output virtual channel head address to the receiver buffer at the old output virtual channel head address as suggested by Traw. The motivation is that (as suggested by Traw, column 6 lines 12-24) the linked list data consists of pointers and cell counts; the cell bodies are stored separately in the Dual Port Reassembly Buffer; the cell body data is separated

from the linked list data to minimize the amount of data movement; this separation is particularly important because it allows linked list management and data movement operations to be carried out in parallel; the separation of the linked list and cell body data will become even more important as the network speed is increased because it limits the degree to which memory bandwidth limitations can throttle the interface. Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces/market place incentives if the variations are predictable to one of ordinary skill in the art.

Regarding claims 5 and 14, Oberman teaches comprising writing a data block to the receiver buffer and reading a data block from the receiver buffer in a single read/write cycle (see Oberman col. 11 lines 41-47).

Regarding claims 6 and 15, Oberman teaches further comprising anticipating the write of a data block to the receiver buffer in a subsequent read/write cycle by reading a new free linked list head address from the receiver buffer an old free linked list head address in a current read/write cycle (see Oberman col. 14 lines 52-55).

Regarding claims 8 and 17, Oberman teaches further comprising supporting a plurality of input virtual channel linked lists, wherein each input virtual channel linked list corresponds to a respective input virtual channel (see Oberman col. 9 lines 44-45).

Regarding claims 9 and 19, Oberman teaches further comprising supporting a free linked list that includes a plurality of vacant data blocks of the receiver buffer (see Oberman col. 7 lines 30).

Regarding claim 10, Oberman teaches further comprising maintaining a mapping indicating a relationship between a plurality of input virtual channels and a plurality of output virtual channels (see Oberman col. 8 lines 36-40).

Regarding claim 11, Oberman discloses dynmic selection of lowest latency path in a network switch (see Oberman col. 2 lines 29 - 58) comprising: • receiving a data block at a receiver of the host device (see Oberman col. 2 lines 59-62), the data block received via an input virtual channel (see Oberman col. 8 lines 11-15); • storing the data block in a receiver buffer (see Oberman col. 7 lines 32-35); • when the input virtual channel has identified therewith an output virtual channel updating an output virtual channel linked list corresponding to the output virtual channel to include the data block (see Oberman col. 8 lines 11-15); and • when the input virtual channel has not identified therewith an output virtual channel (see Oberman col. 9 line 25-26): o updating an input virtual channel linked list corresponding to the input virtual channel to include the data block (see Oberman col. 7 lines 29-57); o processing the data block to determine an output virtual channel for the data block (see Oberman col. 9 line 25-26); o updating an output virtual channel linked list corresponding to the output virtual channel to include the data block (see Oberman col. 9 line 25-67); and o updating the input virtual channel linked list to remove the data block (see Oberman col. 9 line 65).

Oberman implicitly teaches updating an input virtual channel linked list corresponding to the input virtual channel to include the data block but does not explicitly teach updating an input virtual channel linked list corresponding to the input virtual channel to include the data block; storing the data block in the receiver buffer

includes storing the data block in the receiver buffer at an old free linked list head address ; the host device via the output virtual channel, wherein transferring the data block from the input virtual channel linked list of the receiver buffer to a destination within the host device via the output virtual channel includes reading the data block from the receiver buffer at an old input virtual channel linked list head address.

Traw in the same or similar field of endeavor teaches updating an input virtual channel linked list corresponding to the input virtual channel to include the data block; storing the data block in the receiver buffer includes storing the data block in the receiver buffer at an old free linked list head address (columns 5-6, lines 65-11, the use of linked list data structures allows memory to be dynamically assigned to incoming ATM cells and provides a first in-first out (FIFO) queue for the cells of a particular VCI. The memory allocation is dynamic because incoming cells are placed into nodes that are removed from a free list (discussed below) and added to the linked list for the particular VCI. In this way, VCIs that are more active are allotted more memory than less active VCIs. This is accomplished by transferring the node back to the free list. The FIFO queuing characteristic of the linked list structure is important because the order of cells of a particular VCI must be maintained); the host device via the output virtual channel, wherein transferring the data block from the input virtual channel linked list of the receiver buffer to a destination within the host device via the output virtual channel includes reading the data block from the receiver buffer at an old input virtual channel linked list head address (column 7 lines 34-58, the Linked List Controller 18 moves to the initiate command state (S33) when commands requesting operation on

the linked list of a particular VCI are passed from the VCI Lookup Controller 52 (FIG. 5). Those commands include commands for removing a cell, writing (i.e., adding) a cell and reading a cell from a linked list for a particular VCI. The Linked List Controller 18 moves to state S34 when a remove command is issued In state S34 the linked list associated with the VCI to be removed is appended to the free list. The Linked List Controller 18 moves to state S35 when a read command is issued. In state S35 the Linked List Controller 18 locates the first cell reference structure (oldest cell) in the linked list associated with the VCI and determines the location of the cell body in the Reassembly Memory 44. The location of the cell body is passed to the Dual Port Controller 46 (FIG. 4) in state S39 and the count (number of cells in the VCI's linked list) is passed to the host 4 in state S40). Traw further teaches (column 6 lines 56-67, the free list is a linked list of nodes that are not currently assigned to a VCI. The free list is defined by the values stored in locations 2048 and 2049 of the Pointer Table memory 30 (FIG. 3B), which represent the addresses of the first and last free nodes in the free node linked list. Four operations may be performed by the Linked List Manager 10 after initialization: 1. Cells may be added to the linked list for a particular VCI. 2. Cells may be removed from a VCI's linked list. 3. A VCI may be cleared, returning all of its nodes to the free list. 4. A cell count for a VCI may be obtained and returned to the host. The first two operations pass the Reassembly Memory 44 address of the cell body that is to be affected to the Dual Port Controller 46 (44 and 46 part of the Dual Port Reassembly Buffer 12 and are shown in FIG. 4). The count and linked list associated with that VCI are also updated.

It would have been obvious to one having ordinary skill in the art at the time the invention was made to incorporate in Oberman's system/method the steps of updating an input virtual channel linked list corresponding to the input virtual channel to include the data block; storing the data block in the receiver buffer includes storing the data block in the receiver buffer at an old free linked list head address ; the host device via the output virtual channel, wherein transferring the data block from the input virtual channel linked list of the receiver buffer to a destination within the host device via the output virtual channel includes reading the data block from the receiver buffer at an old input virtual channel linked list head address as suggested by Traw. The motivation is that (as suggested by Traw, column 6 lines 12-24) the linked list data consists of pointers and cell counts; the cell bodies are stored separately in the Dual Port Reassembly Buffer; the cell body data is separated from the linked list data to minimize the amount of data movement; this separation is particularly important because it allows linked list management and data movement operations to be carried out in parallel; the separation of the linked list and cell body data will become even more important as the network speed is increased because it limits the degree to which memory bandwidth limitations can throttle the interface. Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces/market place incentives if the variations are predictable to one of ordinary skill in the art.

Regarding claim 12, Oberman teaches further comprising: • transferring the data block from the receiver buffer to a destination within the host device based upon a

corresponding output virtual channel (see Oberman col. 9 lines 25-65); and • updating the output virtual channel linked list to remove the data block (see Oberman col. 9 lines 65-67).

Regarding claim 18, Oberman teaches further comprising supporting a plurality of output virtual channel lined lists, wherein each output virtual channel linked list corresponds to a respective output virtual channel (see Oberman col. 23 lines 17-35).

Regarding claim 20, Oberman discloses dynmic selection of lowest latency path in a network switch (see Oberman col. 2 lines 29 - 58) comprising: • an input that receives data blocks corresponding to a plurality of input virtual channels (see Oberman col. 2 lines 59-62); • a routing module that determines an output virtual channel for data blocks based upon their respective input virtual channels (see Oberman col. 8 lines 11-15); • a receiver buffer operable to instantiate an input virtual channel linked list for storing data blocks on an input virtual channel basis and to instantiate a free list that identifies free data locations (see Oberman col. 7 lines 29-57); • a linked list control module (see Oberman figure 1 box 404 cluster link memory) operably coupled to the receiver buffer (see Oberman figure 1 box 402 input FIFO); • free linked list registers (see Oberman figure 1 box 406 packet free queue) operably coupled to the linked list control module (see Oberman figure 1 box 404 cluster link memory).

Oberman implicitly teaches updating an input virtual channel linked list corresponding to the input virtual channel to include the data block but does not explicitly teach updating an input virtual channel linked list corresponding to the input

virtual channel to include the data block; storing the data block in the receiver buffer includes storing the data block in the receiver buffer at an old free linked list head address ; the host device via the output virtual channel, wherein transferring the data block from the input virtual channel linked list of the receiver buffer to a destination within the host device via the output virtual channel includes reading the data block from the receiver buffer at an old input virtual channel linked list head address.

Traw in the same or similar field of endeavor teaches updating an input virtual channel linked list corresponding to the input virtual channel to include the data block; storing the data block in the receiver buffer includes storing the data block in the receiver buffer at an old free linked list head address (columns 5-6, lines 65-11, the use of linked list data structures allows memory to be dynamically assigned to incoming ATM cells and provides a first in-first out (FIFO) queue for the cells of a particular VCI. The memory allocation is dynamic because incoming cells are placed into nodes that are removed from a free list (discussed below) and added to the linked list for the particular VCI. In this way, VCIs that are more active are allotted more memory than less active VCIs. This is accomplished by transferring the node back to the free list. The FIFO queuing characteristic of the linked list structure is important because the order of cells of a particular VCI must be maintained); the host device via the output virtual channel, wherein transferring the data block from the input virtual channel linked list of the receiver buffer to a destination within the host device via the output virtual channel includes reading the data block from the receiver buffer at an old input virtual channel linked list head address (column 7 lines 34-58, the Linked List Controller 18

moves to the initiate command state (S33) when commands requesting operation on the linked list of a particular VCI are passed from the VCI Lookup Controller 52 (FIG. 5). Those commands include commands for removing a cell, writing (i.e., adding) a cell and reading a cell from a linked list for a particular VCI. The Linked List Controller 18 moves to state S34 when a remove command is issued In state S34 the linked list associated with the VCI to be removed is appended to the free list. The Linked List Controller 18 moves to state S35 when a read command is issued. In state S35 the Linked List Controller 18 locates the first cell reference structure (oldest cell) in the linked list associated with the VCI and determines the location of the cell body in the Reassembly Memory 44. The location of the cell body is passed to the Dual Port Controller 46 (FIG. 4) in state S39 and the count (number of cells in the VCI's linked list) is passed to the host 4 in state S40). Traw further teaches (column 6 lines 56-67, the free list is a linked list of nodes that are not currently assigned to a VCI. The free list is defined by the values stored in locations 2048 and 2049 of the Pointer Table memory 30 (FIG. 3B), which represent the addresses of the first and last free nodes in the free node linked list. Four operations may be performed by the Linked List Manager 10 after initialization: 1. Cells may be added to the linked list for a particular VCI. 2. Cells may be removed from a VCI's linked list. 3. A VCI may be cleared, returning all of its nodes to the free list. 4. A cell count for a VCI may be obtained and returned to the host. The first two operations pass the Reassembly Memory 44 address of the cell body that is to be affected to the Dual Port Controller 46 (44 and 46 part of the Dual Port Reassembly Buffer 12 and are shown in FIG. 4). The count and linked list associated with that VCI are also updated.

It would have been obvious to one having ordinary skill in the art at the time the invention was made to incorporate in Oberman's system/method the steps of updating an input virtual channel linked list corresponding to the input virtual channel to include the data block; storing the data block in the receiver buffer includes storing the data block in the receiver buffer at an old free linked list head address ; the host device via the output virtual channel, wherein transferring the data block from the input virtual channel linked list of the receiver buffer to a destination within the host device via the output virtual channel includes reading the data block from the receiver buffer at an old input virtual channel linked list head address as suggested by Traw. The motivation is that (as suggested by Traw, column 6 lines 12-24) the linked list data consists of pointers and cell counts; the cell bodies are stored separately in the Dual Port Reassembly Buffer; the cell body data is separated from the linked list data to minimize the amount of data movement; this separation is particularly important because it allows linked list management and data movement operations to be carried out in parallel; the separation of the linked list and cell body data will become even more important as the network speed is increased because it limits the degree to which memory bandwidth limitations can throttle the interface. Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces/market place incentives if the variations are predictable to one of ordinary skill in the art.

Regarding claim 21, Oberman teaches further comprising an output athat transmits data blocks corresponding to a plurality of input virtual channels (see Oberman figure 18 and col. 23 lines 17-35 and col. 9 lines 18-64).

Regarding claim 22, Oberman teaches wherein: • the receiver buffer is further operable to instantiate an output virtual channel linked list for storing data blocks on an output virtual channel basis (see Oberman figure 18 ref462 and col. 23 lines 17-35); and • the system further comprises output virtual channel linked list registers operably coupled to the linked list control module (see Oberman figure 18 ref 462) and an input virtual channel to output virtual channel map (see Oberman col. 8 lines 11-15).

Regarding claim 23, Oberman teaches the receiver buffer comprises: • a pointer memory (see Oberman figure 1 box 404 cluster link memory); and • a data memory, wherein a single address addresses corresponding locations of the pointer memory and of the data memory (see Oberman figure 1 box 466 packet link memory).

Regarding claim 24, Oberman teaches the receiver buffer further comprises a packet status memory, wherein a single address addresses corresponding locations of the pointer memory, the data memory (see Calamvokis col. 4 lines 6-14), and the packet status memory (see Oberman figure 1 box 406 packet free queue).

Regarding claim 25, Oberman teaches further comprising a pointer memory read port (see Oberman figure 6 ref 478 sfreadwordptr), a pointer memory write port (see Oberman figure 6 ref 476 writewordptr), a data memory read port (see Oberman col. 8 line 35), and a data memory write port (see Oberman col.8 line 34), each of which can

access the receiver buffer in a common read/write cycle (see Oberman col. 7 lines 37-39).

Regarding claim 26, Oberman teaches wherein: a single pointer memory location can be read from and written to in a common read/write cycle (see Oberman col. 11 lines 41-47); and a single data memory location can be read from and written to in a common read/write cycle (see Oberman col. 7 lines 37-39).

Regarding claim 27, Oberman teaches wherein the receiver buffer comprises: • a pointer memory (see Oberman figure 6 ref 478 sfreadwordptr); • a data memory (see Oberman col. 8 lines 34-35); • a packet status memory (see Oberman figure 3 packet descriptor memory); and • wherein a single address addresses corresponding locations of the pointer memory, the data memory, and the packet status memory (see Oberman col. 7 lines 21-28).

Regarding claim 28, Oberman teaches further comprising: • a pointer memory read port (see Oberman figure 6 ref 478 sfreadwordptr); • a pointer memory write port (see Oberman figure 6 ref 476 writewordptr); • a data memory read port (see Oberman col. 8 line 35); • a data memory write port (see Oberman col. 8 line 33); • a packet status memory read port (see Oberman figure 3 free remove/add ptr); and • a packet status memory write port (see Oberman figure 3 free remove/add ptr).

Regarding claim 29, Oberman teaches wherein: • a single pointer memory location can be read from and written to in a common read/write cycle (see Oberman figure 6); • a single data memory location can be read from and written to in a common read/write cycle (see Oberman col. 8 lines 34-35); and • a single packet status memory

location can be read from and written to in a common read/write cycle (see Oberman figure 3 free remove/add ptr).

Response to Arguments

Applicant's arguments see pages 11-14 of the Remarks section, filed 4/17/2009, with respect to the rejections of the claims have been fully considered and are moot in view of new ground of rejections presented in this office action.

Examiner's Note: Examiner has cited particular columns, line numbers and/or paragraphs in the references applied to the claims above for the convenience of the applicant. Although the specified citations are representative of the teachings of the art and are applied to specific limitations within the individual claim, other passages and figures may apply as well. It is respectfully requested from the applicant in preparing responses, to fully consider the references in entirety as potentially teaching all or part of the claimed invention, as well as the context of the passage as taught by the prior art or disclosed by the Examiner.

In the case of amending the claimed invention, Applicant is respectfully requested to indicate the portion(s) of the specification which dictate(s) the structure relied on for proper interpretation and also to verify and ascertain the metes and bounds of the claimed invention.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to SALMAN AHMED whose telephone number is (571)272-8307. The examiner can normally be reached on 9:00 am - 5:30 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Ayaz Sheikh can be reached on (571)272-3795. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Salman Ahmed/

Examiner, Art Unit 2419